(2)

DTIC FILE COPY

AD-A228 826

SDRL Q14-02021-A

Q14 - Standards Development Plan

Information Resources Dictionary System (IRDS): Analysis and Recommendations

Prepared for
Software Technology for Adaptable
Reliable Systems (STARS)
Unisys STARSCenter
Shipboard and Ground Systems Group
Reston, Virginia

Prepared by
Unisys Corporation
Defense Systems
System Development Group
P.O. Box 517, Paoli, PA 19301-0517

Contract No. F19628-88-D-0031
IDWO P.O. 010412

April 7, 1989

Contents

1 Executive Summary

An Information Resources Dictionary System (IRDS) provides an inventory system for the information environment of an enterprise and is intended to be the principal tool for managing the information resources of an enterprise. Its strongest advocates proclaim than an IRDS is central to a software engineering environment and that its uses include the modeling of the software engineering lifecycle and configuration control through the development, integration, and execution environments.

The STARS Interface Standards Task has investigated the IRDS standards and their meaning to STARS. This paper describes the IRDS and the current status of IRDS standardization efforts. It reports on early, federally funded users of the IRDS standard and on research which involves the intersection of Ada Programming Support Environments (APSE) and the IRDS standard. The original work within this report is the use of recently published guidelines for the comparison of semantic data models to analyze the models of CAIS-A, the ANSI IRDS standard, and the STARS EOM.

The results of this study are a set of conclusions concerning the importance of the IRDS standards to STARS. In general, this paper is concerned with the role STARS should play as an Ada advocate within the ANSI IRDS standards committee, the use of CAIS-A nodes for an IRDS implementation, and the relationship between IRDS interfaces and the STARS Environment Object Manager (EOM).

We conclude:

- Although there are competing proposals within the American and international IRDS standards groups, there are signs that eventually a single world-wide standard will be adopted.
- Ada package specifications for the IRDS standard would not only provide access to IRDSs, they can, directly or indirectly, provide access to relational databases.
- STARS should take a wait and see attitude toward the development of Ada/IRDS bindings. STARS should provide low-level support for Ada/IRDS standardization through development of, and participation in, an Ada/IRDS interest group.
- The study which is presented in the Appendix to this paper, raises issues concerning the granularity of objects which would be stored in the dictionary databases.
- The EOM model is more powerful than the IRDS model and it is in line with NASA researcher's recommendations for the modeling of objects and phases in the software engineering lifecycle. STARS EOM research might eventually feed into a second generation IRDS standard based on objects.

2 Background

Initially the Information Resources Dictionary System (IRDS) was developed for passive description of the metadata of an enterprise. The metadata would describe the information resources of the enterprise: its users, processes, data, and so forth. The National Bureau of Standards initiated work on a Dictionary Standard in 1980. This was before the development of CASE tools, before *object management* became a buzzword, and certainly without considering the developing CAIS. Most of the IRDS literature, published by NIST, reflects this original vision of the Dictionary System.

The IRDS seems to be many things to many people. Descriptions of the IRDS seem to vary depending upon the issues of the describer. IBM and its CASE vendors describe a storehouse or repository which will provide data interchange among CASE tools and access to application databases. Space station researchers emphasize the modeling power of entity-relationship models for organizing system resources, and for the enforcement of precise, abstract interface specifications between life cycle phases. Within the ANSI committee there is interest in solving problems of model integration and using the IRDS within distributed environments. Federal users envision MIS support ranging from a simple passive dictionary containing metadata about an enterprise's information resources to the ambitious Army Data Encyclopedia project.

Within STARS, the Boeing Q24 Object Management Task has used the IRDS command language as a model for the object definition language and object manipulation language of the STARS Environment Object Manager (EOM). The author of this paper points out that a services interface to the IRDS offers the Ada community an alternative to SQL as an interface to relational databases.

So the IRDS, which was not developed for the purpose of CASE interchange, nor for the purpose of providing object definition or object manipulation languages, nor for the purpose of providing schema integration for distributed databases, and without regard for Ada Programming Support Environments, or providing a new way for Ada to access relational databases is now viewed in all of these contexts. These visions are overlapping, not contradictory.

The first subsection describes the features of the IRDS, the second summarizes the status of various standardization efforts.

2.1 Features of the IRDS

A short paper based on the November, 1988 Keynote address at the Seventh International Conference on Entity-Relationship Approach provides an excellent, concise overview of the history, features, and uses of the IRDS [Winkler 88].

The IRDS *data levels* are shown in Figure 1. In this layered architecture each level defines and describes the level below it. The levels are: 1) the fundamental level, 2) the Information Resources Dictionary (IRD) definition level, 3) the IRD level, and 4) the application level. The fundamental level is fixed by the model of the standard. The ANSI standard, X3-138 defines a required "minimal IRD Schema" at the IRD definition level. X3-138 also defines an optional "functional IRD Schema" which can be provided by implementers and adopted by users. However, most IRDS users will define an IRD Schema appropriate to their enterprise, making use of the IRD fundamental level. Thus the IRDS is *self descriptive* and *extensible*. Extensibility is a two edged sword. It allows the development of IRD definitions appropriate to the application, but it also raises portability problems. IRDSs with different IRD Schemas will be incompatible. The model integration problem will apply to both the IRD and IRD definition levels.

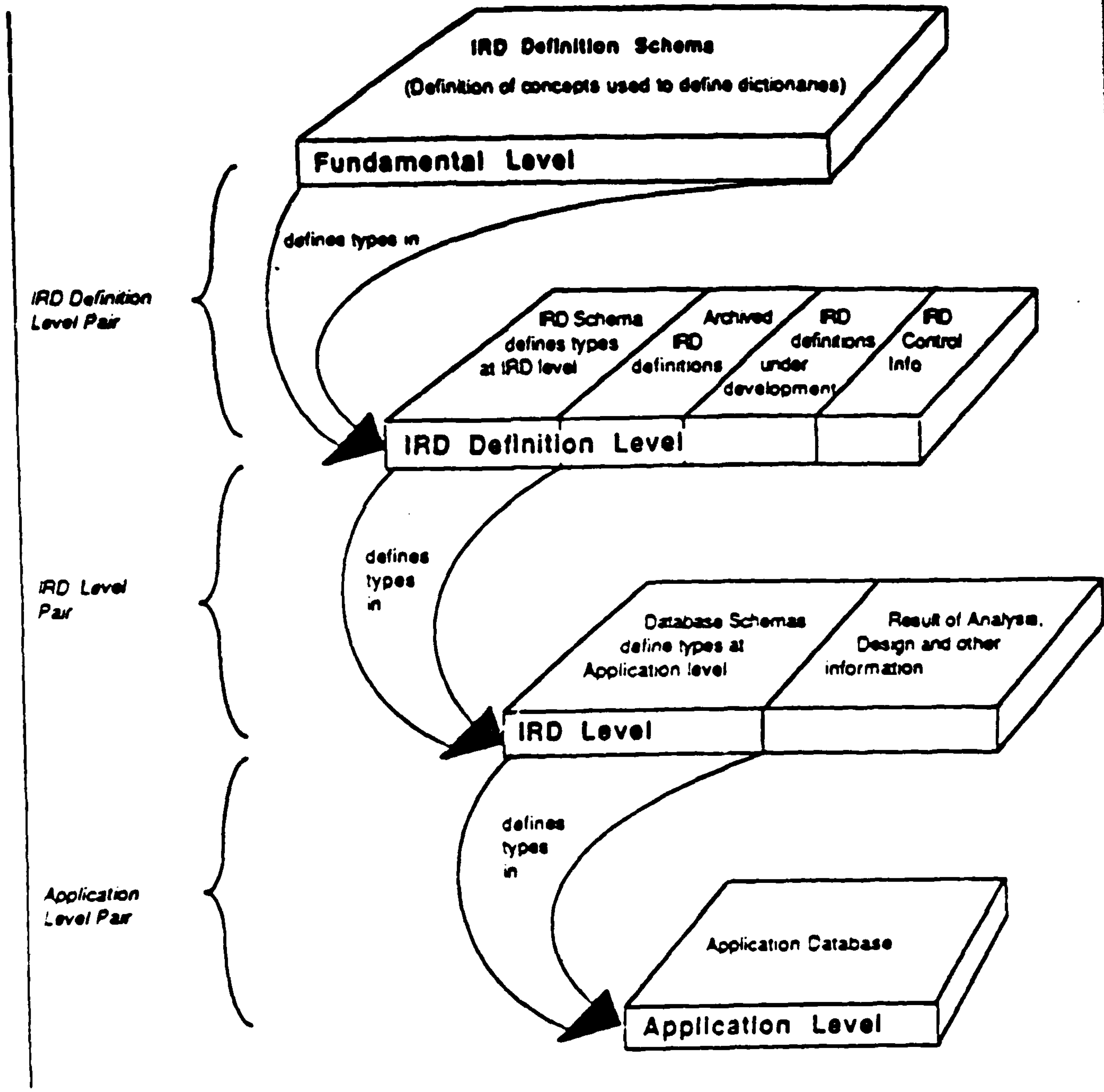

Figure 1: ISO Figure showing IRDS levels

The IRDS *framework* is shown in Figure 2. The framework [SC21N2642 88], also known as the reference model, shows that the services interface is the central specification, which will be used by the front-end tools and applications to describe, retrieve, and manipulate two, or possibly three levels of data: the IRD, the IRD schema, and (with a directory system) the application database.

The services interface corresponds to an Ada abstract interface. Ada language bindings to the interface would be package specifications.

Application programs and man-machine interfaces as well as the command language and panel interfaces would be built on top of the services interface. Man-machine interfaces to the services interface provide a mechanism for supplementing (graphical interfaces) and ignoring (other panels, other languages) the command language and panel interfaces which have been standardized by X3-138 [ANSI 88]. X3-138 is a stand-alone IRDS standard, without a services interface. It is the only IRDS standard which has been adopted, either nationally or internationally.

Modularity is another feature of X3-138. The standard has a number of optional modules. Implementations of the standard will differ in that different sets of modules will be implemented. This will be another source of portability problems.

Four *control facilities* are required in X3-138: versioning, life cycle phases, quality-indicators and views. X3-138's optional modules include the IRDS security module and the extensible life cycle phase facility. A system may have several *uncontrolled* life cycle phases generally representing non-operational stages of a system life cycle such as specification, design, or development. The single *controlled* life cycle phase is for entities in the IRD that describe data in operational systems. Additionally, there is a single *archived* life cycle phase.

Last, but certainly not least, the IRDS provides entity-relationship (E-R) modeling. The ANSI E-R approach provides for definition and manipulation of dictionary metadata which would describe both SQL and NDL databases. The ISO E-R model is closely tied to SQL. International disagreement over the data model has slowed down IRDS standardization of both the international and American standards.

2.2 Standardization of the IRDS

2.2.1 History of IRDS standardization

Advocacy is a critical aspect of standards development. One question for our study of the IRDS Standard has been to consider the importance of the standard for Ada software engineering and whether to recommend STARS funding of Ada advocacy within the IRDS committee.

The National Institute of Standards and Technology (NIST nee NBS) has been the primary advocate of IRDS standardization since 1980. The references listed are only a subset of the NBS/NIST publications. [Goldfine 88c,Law 88,Goldfine 88b,Goldfine 88a,Newton 87]

Another important advocate is AOG Systems, a company that developed the ANSI IRDS standard, X3-138, under contract to NIST. The standard was adopted by ANSI in October, 1988. It will become a FIPS sometime this year. The developer of the ANSI proposal for a services interface

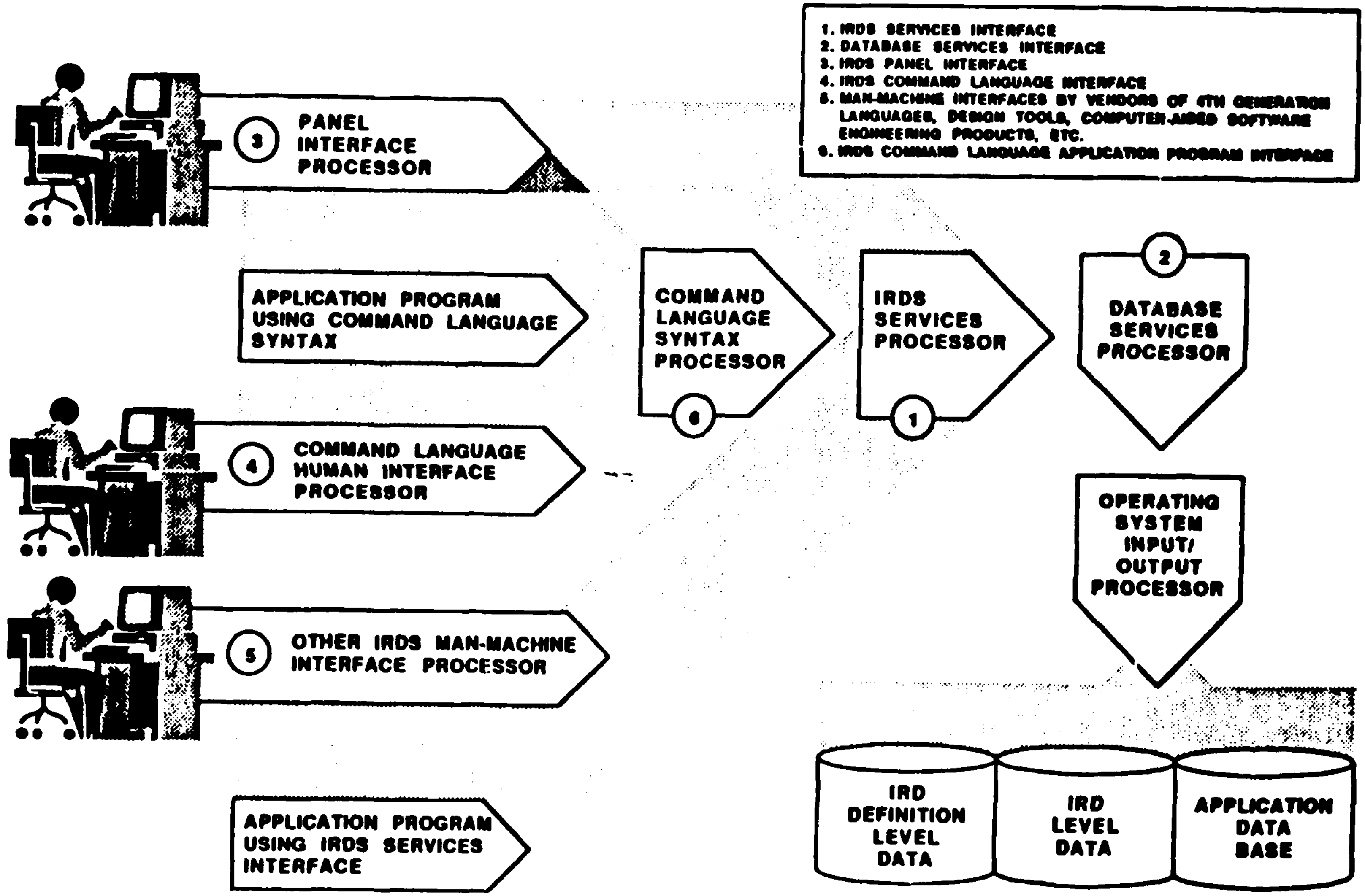

Figure 2 Major IRDS Interfaces

from [winkler 88]

[X3H4 88] is Pansophic Systems. This advocacy was financed privately by the company which has developed a commercial implementation of X3-138 where the command and panel interfaces use this services interface.

The ISO standardization effort started with an ANSI base document. ISO proceeded with a draft proposed standard for the framework. ISO does not intend to standardize command language or panel interfaces. The developing ISO and ANSI standards were compatible until October, 1987, when the British Standards Institute proposed a different flavor of E-R model for the IRDS. The specter of incompatible ANSI and ISO standards has slowed progress on the IRDS standardization efforts.

Last fall, IBM began lobbying, nationally and internationally, for acceptance of the interface specifications to the IBM *repository* as the IRDS services interface standard. In January, the ANSI committee agreed that IBM should bring its base document to its April meeting. Meanwhile, the conflicting ANSI and ISO proposals are proceeding through the standardization process at a reduced pace.

Following standardization there must be conformance testing. This is a serious and difficult issue for NIST and it has not yet been addressed. As reported in Section 3.2, one federal user is already planning an implementation of the services interface without conformance to X3-138. While the ANSI services interface will augment the X3-138 standard, conformance to X3.138 will not be required to claim conformance to the ANSI services interface.

2.2.2 Three Proposals for a Services Interface

ISO is developing a services interface [SC21WG3N669 88] using its data model. The ISO version of the services interface is closely tied to SQL, and it could provide an SQL data definition language. Technical and political issues combine to make the ANSI standard the one of interest to STARS.

The ANSI proposal (the Pansophic services interface) is less closely tied to SQL. Only Pascal language bindings have been specified, but development of Ada language bindings would be relatively straightforward. Both the ISO and ANSI services interfaces provide a complete set of error messages. Thus an Ada/IRDS services interfaces can be used as a front end to relational DBMSs with consistent package specifications for error handling, data definition, and data manipulations.

However, this is a step back from a declarative relational language such as SQL. IBM claims that its proposal provides an interface that requires little or no navigation through the E-R network to be performed by the tool developer. IBM recommends a *physical view* which is implementer defined, a *conceptual view* which is an E-R semantic network, and a *tool view* which is a hierarchical view of the conceptual view's E-R network. The IBM proposal suggests template trees which are storable, sharable objects which define the external view, data processing order, and entity precedence for calls to the services interface.

The IBM repository is a part of the IBM Systems Applications Architecture [SAA 88]. An IBM spokesman discussing standard interface specifications to the IBM repository has been quoted as saying, "As the standard evolves, we expect our standard to comply." That IBM understatement is an interesting way of putting it since IBM is working, both nationally and internationally, to move the standard bodies toward compliance with IBM. A mapping from the ISO conceptual model to the tool view might serve to resolve the data model differences between ISO and ANSI.

2.2.3 Future direction of the IRDS and other standards

There are other activities within X3H4 besides work on the services interface. Some of these efforts are toward standards which would augment the X3-138 standard. These include the development of an export/import facility, the reference model document, and a naming convention verification module. Other standardization efforts are aimed toward the development of a second generation IRDS standard. An IRDS with n-ary relationships, a thesaurus view of the IRDS, an IRDS in distributed environments, and an object-oriented IRDS have all been mentioned in the context of "IRDS-2."

Addressing the problem of model integration is a priority for many of the X3H4 members. A proposal for model integration includes a detailed presentation of the issues with a model integration bibliography pointing to original work and a survey of schema integration methodologies[X3H489016 , Batini 86]. The proposed work item is the development of a technical report which would outline the state of the art and specify the minimum functionality that would be required for a tool that purported to provide computer-aided support for the model integration process. The results of the technical report would be the basis for further standardization efforts.

There are other standards groups working on communications between CASE products, conceptual schema models, and distributed processing. There is some overlap and potential contention between X3H4 and X3T2, the committee on Data Interchange. X3T2 has a new *Conceptual Schema Project.* Also, there is potential overlap between X3H4 and X3T5, the committee on OSI architecture, and with the work on Open Distributed Processing. The X3/DBSSG (Database Systems Study Group) is trying to get standards that fit together, or at least which do not conflict.

The EDIF/CASE group is a standards group, independent of ANSI, which is addressing issues of data interchange among CASE tools. There is an overlap between the IRDS and the EDIF/CASE efforts. However, leadership in the EDIF/CASE committee concurs with the view that in the long run the repository approach of the IRDS will be more useful than the direct tool-to-tool standard being developed by EDIF/CASE.

3 Experience with the Standard

While most Federal users will probably wait for the appearance of commercial implementations of the standard, some federally funded implementation have been developed. NIST implemented a public domain version, in C, using Oracle. Three contractors are using implementations of the IRDS in connection with research on the Army Data Encyclopedia (ADE) Project. Argonne Laboratories has developed an Ingres implementation of the IRDS and is definitely moving ahead with a customized implementation of the IRDS for their work. The Argonne experience makes the point that, at the same time that an IRDS can be essential, the existing IRDS standard is worrisome.

3.1 Army Data Encyclopedia

The Army Data Encyclopedia (ADE) is a mammoth undertaking[LBL 88]. Figure 3 shows that complexity and that the IRDS is the central component. Lawrence Berkeley Laboratories (LBL), Honeywell, and American Management Systems (AMS) are the three contractors that have worked on this project.

LBL, which has worked on the ADE architecture has also done research on a thesaurus [McCarthy 88] which would include capabilities to:

- Manage *definitions* and *cross-references*;
- Reconcile diverse *nomenclatures* and *classifications* from multiple sources;
- Enforce data and metadata *integrity constraints* in an active framework;
- Generate *dynamic menus* and expand user queries;
- Link information between different types of entities and databases.

Honeywell's Army Encyclopedia work is called ANSWER: Army's Nonprogrammer System for Working Encyclopedia Requests[Dwyer 88]. This system provides support for the encyclopedia with a set of tools that provide enterprise management, database registration, browsing, query formulation, and distributed query processing. The IRDS is the storage mechanism for the encyclopedia information. A prototype is expected in May or June of 1989. Additionally, Honeywell is using an IRDS internally, adding capabilities to generate data base translators based on definitions of source/target mappings entered into the IRDS system.

AMS's findings on the Ada impact on the Army Data Encyclopedia are reported in Section 4.2.

3.2 Argonne Laboratories

Argonne Laboratories does work for J8 (the Force Structure, Resource, and Assessment Directorate of the Joint Chiefs of Staff) [Robinson 88]. J8 runs a large variety of simulation models. These are very large, 18-20 year-old FORTRAN programs, requiring large amounts of data which is collected separately for each simulation. The IRDS is essential for modernization. They need to establish an inventory of the data which they have and eliminate multiple requests for the same data. Argonne has built an Ingres implementation of the X3-138 standard.

Argonne is moving ahead with the development of an IRDS which will be implemented on Ingres, a procurement requirement. This will be an active IRDS, with a services interface through which updates may be introduced and which will be able to construct screens of data on the fly. In addition to the dictionary database, a second database will provide a directory to individual simulation model databases. A decision will be made after the April X3H4 meeting whether to implement the IBM or the Pansophic services interface.

The Argonne implementation of IRDS will not conform to the X3-138 standard. This is because provisions for life cycle management cannot satisfy MIL-STD 2167, and the module for security is insufficient for the federal multi-level security guidelines.

This is a serious criticism of the X3-138 standard. There will be severe problems in implementation, conformance, and conformance testing of the 800 page X3-138 standard. It is important to take note of the fact that while the services interface standard augments the X3-138 standard, conformance to X3.138 is not required to claim conformance to the services interface.

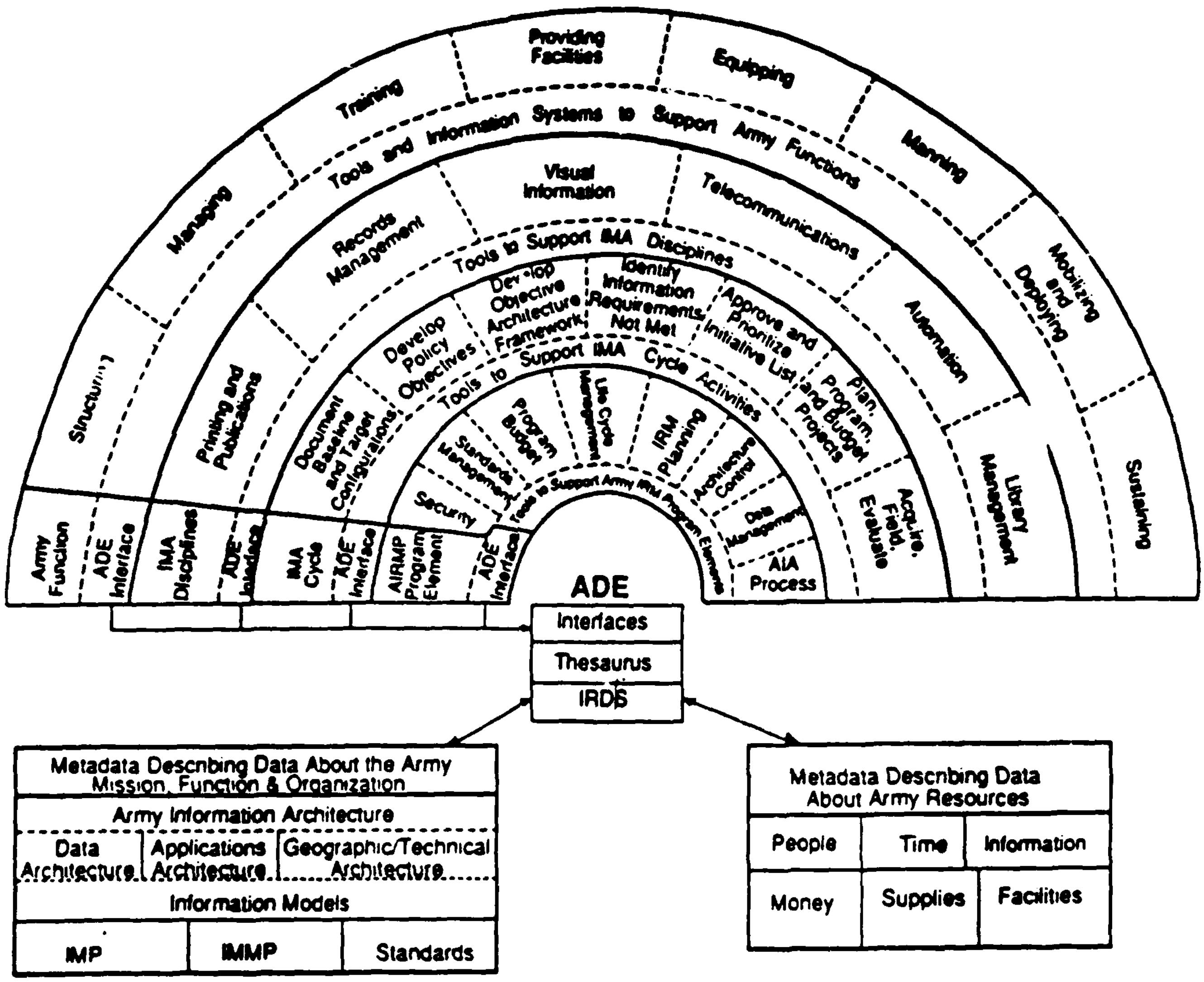

Figure 3: LBL Figure showing ADE Reference Model

4 IRDS and Ada Programming Support Environments (APSE)

This section points to research which has delved into the relationship between an APSE and the IRDS.

4.1 Modeling the Space Station Program

Dr. Charles McKay heads two research groups at the University of Houston Clearlake. They are both concerned with Ada Software Engineering Environments. One is a NASA research laboratory for the Space Station Program; the other laboratory does commercial work. The IRDS has been described as the "focal point of the design" for the Houston work and also for related commercial work in West Virginia which is based on Dr. McKay's design. The general themes of the Houston documents are expressed in the title of Dr. McKay's chapter in the *Ada Reusability Guidelines*: "Conceptual and Implementation Models Which Support Life Cycle Reusability of Processes and Products in Computer Systems and Software Engineering"[McKay 89]. The Parnas objects (modules which reflect design decisions) which are of interest to Dr. McKay are often themselves models (hence the "taxonomy of taxonomies" phrase which is used in Houston).

Ideas expressed throughout the Houston documents [McKay 88c,McKay 87,McKay 88b,McKay 88a] indicate a vision of the use of the IRDS throughout the software engineering life cycle, using objects, sets of objects, and objects which are themselves models. Stable Interface Sets and Stable Frameworks are notions which involve the establishment of Life Cycle Phases, points in the overall lifecycle where products of preceding activities can be precisely defined and specified to determine the completeness of products to that point. As described earlier, life cycle partitions are an important feature of the X3-138 standard. The Houston vision of the IRDS seems to be closely tied to this feature which tends to be forgotten in all the IRDS discussion about competing data models and services interfaces.

4.2 Ada Impact on the Army Data Encyclopedia

The Army Data Encyclopedia (ADE) was described in an earlier section. An ADE contract from the U.S. Army Information Systems Software Center (USAISSC) to American Management Systems produced a document titled "The Ada Environment for Requirements Analysis, Conceptual Design, and Prototyping of the Ada Subset of the Army Data Encyclopedia" [AMS 88]. That document discusses Ada's *object-oriented* design, *object-oriented* Life Cycle, and *object-oriented* methodologies and how they would affect the ADE. [1]

The report concludes that Ada would affect four aspects of the ADE: 1) content and organization, 2) functional capabilities, 3) data administration issues, 4) environmental issues. One data administration issue is the handling of Ada's name overloading feature, given the curent Army data naming standards. In discussing environmental issues, the report states:

[1]The narrower definition of *object-oriented* which is adopted in the Appendix of this paper does not encompass the Ada Programming Language.

The ADE must be available in the crucial design intensive period and must be able to easily interface with design tools. There should be analysis to see if the ADE should be designed using CAIS standards in addition to IRDS standards. For the short term the Army may want to choose a software support tool(s) as a standard for use with Ada until the CAIS interface is available. The ADE requires efficient Ada interfaces to relational DBMSs.

In January there was an interruption of funding for the LBL work on the Army Data Encyclopedia. We do not know the curent prospects and/or timeframe for development of the ADE.

4.3 AOG Studies

In 1983 AOG Systems was funded by AJPO under contract to NBS to report on the applicability of the FIPS Data Dictionary System to APSEs. The conclusion of the report is that the DDS functionality would be an integral part of an APSE to facilitate the integration of system and tool functionality, and to support the analysis design, development, test, integration, and management of software which operates in the APSEs [AOG 83b].

As a part of that 1983 contract an IRDS Ada schema was developed[AOG 83a]. The Ada schema provides the IRDS entities and relationships necessary for the Ada language. It is *not* a set of Ada package specification which provide bindings to a services interface. There are eleven entity types in the schema, such as *package, task-type* and *expression.* The *expression* entity-type is interesting in that it requires extension of the IRDS to n-ary relationship types which are not provided the IRDS standard.

AOG has provided Q14 with two 1985 papers (unsuccessful proposal efforts), which highlight a connection between IRDS and an APSE. One was the bid on the contract to develop CAIS-A. The other is to construct a prototype Ada programmer workbench using the IRDS as a foundation.

AOG's president, the document editor for X3-138, has outlined potential solutions to the problem of adding n-ary relationship to the standard[X3H489155]. This ideas in this 1988 document are described in the appendix, since they are applicable to the data model analysis which is presented there.

5 IRDS, CAIS-A, and the EOM

All current implementations of the IRDS use relational databases to hold the metadata of the Information Resources Dictionary (IRD) and the IRD Schema. The STARS Interface Standards Q14 Task has considered whether an IRDS might be implemented through the CAIS-A node model[CAISA 88]. At the same time that Q14 has been studying the IRDS standard, the Boeing Q24 Task has been concerned with the development of the Environment Object Manager (EOM) for the STARS environment [Q24570 88,Q24590A 89,Q24610B 89,Q24620 89].

The Boeing documents rest on an understanding of the relationship between CAIS-A and the EOM, and their study has been extremely useful in developing an understanding of the relationship between CAIS-A and the IRDS. The approach for development of the EOM on CAIS-A shows how a mapping might be accomplished for implementation of an IRDS database directly on CAIS-A.

A contribution of the Unisys Q14 work on the IRDS is a comparison of the IRDS and CAIS-A, using Peckham and Maryanski's guidelines for comparison of semantic data models. This study is described in detail in Appendix A of this paper. Also, Appendix A includes definitions of ambiguous terms such as "object."

The analysis served to emphasize striking dissimilarities between the IRDS semantic data model and the CAIS-A node model. In the IRDS an entity is an *aggregation* of its attributes, but a CAIS-A node has *contents* and as well as *properties* which are called attributes. While the CAIS-A attributes *can* be used to express aggregation, the use of attributes to provide aggregation (while ignoring node contents) raises important issues of granularity. The dictionary of an IRDS can be quite large and it may consist of many entities of fine granularity. Such a database is appropriately handled by an efficient relational DBMS. We assume that the development of Ada bindings for the IRDS would be undertaken in order for CASE tools to access dictionary data which is stored on relational DBMSs.

The analysis shows that the EOM model is more powerful than the model of the IRDS. The proposed EOM model includes objects, specialization/generalization, n-ary relationships, and unstructured object representation. None of these are available in the IRDS model. Although the diagrammatic representation of the model does not refer to the contents of objects, other Q24 design documents do suggest operations on documents, objects of coarse granualarity which are the contents of a CAIS-A node. NASA researchers discuss the modeling of objects which are coarser grained yet.

6 Conclusions

This section summarizes the support for the conclusions which were given at the beginning of this paper in the Executive Summary.

6.1 The future of IRDS standardization

As of March 20, this observer's opinion is that IBM could succeed, not only within the ANSI committee, but in the international arena as well. IBM is working with CASE tool vendors, and products are currently being developed which will work with the IBM specifications. The IBM standard is expected to become de facto standard in some arenas. The federal government has a large investment in the existing ANSI standard, X3-138, which it has sponsored. Within the IRDS standards community there is a desire for a single world-wide standard. Acceptance of the IBM repository interface standards provides this possibility since IBM is also working on ISO acceptance. On the other hand, the IBM proposal is two years behind the existing X3H4 proposal and there is pressure for a FIPS in the short term.

If the goal of a world-wide IRDS standard is achieved, the standards groups will be able to move forward, address the issues of model integration, and proceed toward the development of a second generation IRDS standard. Conversely, continued difficulties in the current standardization efforts could damage the ANSI committee's ability to proceed with other efforts.

6.2 Ada/IRDS advocacy

The standardization, and implementation, of a world-wide IRDS services interface standard can promote the portability of front-end CASE tools from one environment to another.

Ada package specifications for the IRDS standard would not only provide access to IRDSs, they can, directly or indirectly, provide access to relational databases. Direct access is provided to the databases which normally hold metadata, but could be used to hold any kind of data. With direct access, IRDS interfaces may be viewed as providing an RDBMS front end. The interface has an E-R or hiearchical view of data, and a standard set of error messages which protect the program from the need to interpret the error messages of the RDBMS. Indirect access to tional databases is provided when an IRDS implementation provides a directory, in addition to the dictionary, and allows retrieval of application data held in relational databases.

Advocacy for the IRDS standard is coming from vendors, such as IBM and Pansophic Systems, that develop software primarily for mainframe environments. The fact that an IRDS advocate as powerful as IBM does *not* propose to develop Ada bindings to a standard or to provide implementations which support the Ada language is of serious concern. In order for the IRDS interface to be significant for STARS there will need to be implementations of the IRDS, in a workstation environment, which support the Ada/IRDS bindings. Therefore, Ada/IRDS advocacy must include vendors who intend to develop Ada/IRDS implementations on appropriate platforms. There is commercial Ada work which is IRDS centered in Houston and in West Virginia. Therefore, it is possible that IRDS-based products supporting Ada are planned, however this Ada/IRDS interest has not been expressed in the form of participation in the formal standards committees.

6.3 Architecture

We make a distinction between the IRDS *idea* which definitely is relevant to STARS, and the IRDS *standard* which may or may not evolve in a way that it will be useful for STARS.

There is an overlap between the functionality of IRDS, CAIS-A, and the Environment Object Manager which is being developed by Boeing under Q Task 24. The IRDS is touted as being a central storehouse of information which can be accessed by CASE tools, as the basis for system life cycle and project management, for source and object library management, and so forth. Issues of environment integration and granularity swirl around the IRDS, CAIS-A, and the EOM.

The papers from Univerity of Houston Clearlake describe EA/RA modeling of objects where the granularity is *coarser* than that of CAIS-A nodes, such as phases in the software engineering lifecyle of the space station program. The EOM seems quite appropriate for this type of modeling. On the other hand the work which was done an Ada/IRDS schema suggests the using the IRDS to express relationships between the objects of very *fine* granualarity down, to Ada expressions. This is the level of granularity at which Intermediate Languages, such as DIANA, operate.

All existing implementations of the IRDS make use of RDBMSs to hold the metadata, and they assume that this metadata describes fine grained data which is held in conventional databases. We believe that relevance of the IRDS to STARS is tied, somehow, to the integration of fine grained data into the STARS SEE. If all of the objects in the SEE are managed by the EOM, which has its own dictionary system, then the IRDS probably has no place in the STARS SEE. But integration into a common environment database (or object manager) raises performance issues concerning support of the management of fine-grained objects. If we assume that the SEE will include RDBMSs, then an IRDS might be used to relate objects managed by the object manager, data in a relational DBMS, and DIANA trees as well. This raises another set of issues concerning environment extensibility and the handling of objects which are not integrated through a single object manager.

6.4 Recommendation

The overlap between CAIS-A, the EOM, and the IRDS is such that these interfaces may be competing rather than complementary. While the *idea* of an IRDS will certainly be a part of an APSE, the IRDS *standard* is being developed in another context, not the context of an APSE. Theoretically Ada bindings to the IRDS interface could compete with other approaches to Ada - SQL interfacing, however this is simply a back burner possibility.

STARS work on CAIS-A and the EOM are fundamental, and STARS attention to the IRDS should be limited to monitoring the evolution of the standard.

A Comparison of CAIS, IRDS, and the EOM

Section 5 in the body of this paper gives the rational for, and conclusions from, this study.

A.1 Results

The analysis served to emphasize striking dissimilarities in the models.

- The CAIS-A node model is not a semantic data model. Semantic data models do not have anything analogous to the *contents* of a CAIS node.
- CAIS-A provides for aggregation through structural nodes. The IRDS provides for aggregation through the attributes of an entity. The CAIS-A model is appropriate for managing objects of coarse granularity, and questionable for use with fine grained data .
- The relational semantics of the CAIS and the IRDS are dissimilar. A relationship is CAIS is analogous to a relationship-class in the IRDS. However, CAIS relationships have instances and IRDS relationship-classes do not.
- CAIS provides for enhanced unstructured object representation.
- The proposed EOM model includes objects, specialization/generalization, n-ary relationships, and unstructured object representation. None of these are available in the IRDS model.

A.2 "Object-oriented" and other definitions

One survey, [Peckham 88], considers *object-oriented databases* a subset of *semantic databases* and *objects* the same as *entities.* The other survey, [Hull 87] draws distinctions between these terms. The definitions offered here are based on those provided by Hull and King.

Hull and King contrast semantic data modeling with object oriented programming, semantic networks, and object-oriented databases.

Semantic data model. Semantic data modeling is the modeling of data which is held in a database. The term does *not* include the modeling of in-memory semantic networks. Because semantic networks are generally in-memory tools, the sorts of research efforts that have been directed at efficient implementation of semantic databases have not been applied to them. Neither does the term semantic data model apply to the node model of CAIS-A. The nodes, attributes, and relationships of CAIS-A model neither the data of an enterprise nor the data of software engineering. The CAIS-A model is closely tied to the application for which it was developed, modeling of an operating system. The nodes, attributes, and relationships of CAIS-A are strictly different from the entities, attributes, and relationships in semantic data models.

Semantic model. This is a broader term which encompasses semantic data models, semantic networks, and the CAIS-A node model. When the context is clear, it can be used as shorthand for semantic data model.

Hull and King describe the fundamental distinctions between semantic data models and object-oriented programming:

Essentially, semantic models encapsulate structural aspects of objects, whereas object-oriented languages encapsulate behavioral aspects of objects. There are three principal features of object-oriented languages. The first is the explicit representation of object classes or (types). Objects are identified by surrogates rather than their values. The second features is the encapsulation of "methods" or operations within objects. The final feature of object-oriented languages is the inheritance of methods from one class to another. . . . Object oriented models do not typically embody the rich type constructors of semantic models. From the structural point of view, object-oriented models support only the ability to define single-and multivalued attributes.

This paper adopts a restricted definition of object. The term *object* requires the encapsulation of methods or operations; otherwise *entity* or "object" in quotes will be used. By this definition Ada *does* have objects. A private datatype which can be manipulated only through publicly available procedures or functions is an object. This paper also adopts a restricted definition of *object-oriented.* The modifier object-oriented is reserved for programming languages and databases which provide *both* inheritance and specialization of methods. By this definition Ada is *not* an object-oriented language.

Whereas Peckham and Maryanski consider object-oriented databases a subset of semantic databases, Hull and King draw a distinction:

Object-oriented database models are fundamentally different from semantic (data) models in that they support forms of local behavior in a manner similar to object-oriented programming languages. This means that a database entity may locally encapsulate a complex procedure or function for specifying the calculation of a data operation. This gives the user the capability of expressing, in an elegant fashion, a wider class of derived information than can be expressed in semantic models. . . . On another dimension, object-oriented (data) models are similar to semantic (data) models in that they provide mechnisms for constructing complex data by interrelating objects.

Thus, object-oriented data modeling has the power of semantic modeling and of object oriented programming. A similar modeling combination can be achieved in the implementation of a semantic network with an object-oriented programming language, as was done in PCLnet [Heineman 87]. However, as previously noted, the semantic networks of AI are implemented in memory and avoid the hard problems of semantic databases. Object-oriented databases are truly a confluence of three disciplines: the object-orientation (inheritance as well as incapsulation) of programming languages, the modeling power of semantic networks, and the emphasis on efficiency and security that are required of database management.

A.3 Peckham & Maryanski Analysis of the CAIS-A and IRDS models

This section describes the method of Peckham and Maryanski and uses it to analyze the CAIS-A and IRDS models. The result of the analysis emphasizes that the semantics of the nodes, attributes, and relationships of CAIS-A are quite different from the semantics of entities, attributes, and relationships of IRDS. These semantic differences emphasize that the CAIS-A node model is fundamentally different from a semantic data model.

This section assumes a basic familiarity with both the CAIS-A node model and the IRDS model. Introductions to these models may be found in [Winkler 88,SofTech 88].

Peckham and Maryanski state:

> *Every semantic model has* objects *(or entities),* relationships *(functional or relational),* dynamic properties, *and a means for handling* integrity constraints. *Relationships can be characterized by the abstractions they are capable of representing and the means by which they do so. Dynamic properties can range from the simple specification of insertions and deletion constraints to the modeling of operations and transactions. Constraints can be collected from the user and represented and/or automatically implied by the semantics of the model's relationships. Both the level and mechanisms of information representation are used to characterize and compare models. In this spirit, the following characteristics are identified as being fundamental to semantic data models.*

Peckham and Maryanski proceed to identify and define eight characteristics of semantic data models and then they analyze eight semantic data models in the context of these eight characteristics. The following eight subsections correspond to the eight characteristics used for analysis.

A.3.1 Standard Abstractions Present.

One of the most important characteristics of a model are the abstractions which it provides. The four abstractions discussed in both review articles are *classification, generalization, aggregation, and association.* Briefly, classification provides for instances of a type and generalization provides for subtypes. The distinction between aggregation and association is explained in [Peckham 88]:

> *Although association and aggregation define new object types from previously defined types, they represent fundamentally distinct abstractions. Aggregation provides a means for specifying the attributes of a new object type, whereas association is the mechanism for defining a type whose value will be a set of objects of a particular type.*

The IRDS semantics of an entity with attributes is identical to the *aggregation* abstraction of a relational database, where a table has attributes (fields). However in CAIS-A the attributes do not make up an aggregate which fully define a node. The attributes are properties of a node which can have *contents.* Contents is certainly a distinguished property, unlike anything in semantic data models. The nodes of CAIS-A are fundamentally different from the entities of the IRDS. In CAIS-A aggregation is provided through structural nodes. This differences in the way in which aggregation is provided makes fine grained aggregation "natural" in the IRDS model and coarse grained aggregation natural in the CAIS-A model.

CAIS-A has the *specialization* abstraction. Node-kind-definitions (NKDs) are created by specializing NKDs which have already been established. At first glance, CAIS-A resembles semantic networks which have a taxonomic *generic network* and an *instance network*. The generic network provides generalization/specialization, and the instance network does not. However, CAIS-A uses the term *parent* in an unexpected way. *Parent* does not refer to the superclass/subclass relationship in the generic network. In CAIS-A, *parent* is used to describe an *instance* in the instance network which is the starting point of a path, which is the primary relationship, to another instance node.

A basic E-R model provides only for aggregation. The IRDS model, with its multi-leveled architecture, provides for classification as well. The CAIS-A model provides for classification and generalization. CAIS-A node model does *not* provide aggregation, an abstraction which is provided by all semantic data models.

Examples of semantic data networks with the *association* abstraction are described in [Peckham 88]. Based on that discussion, association does not appear in either the IRDS or CAIS-A.

A.3.2 Unstructured object representation

This characteristic of a model refers to the handling of primitive types such as strings, integers, and reals, generally supported by the underlying hardware. Most semantic data models have *limited* unstructured object representation. The Semantic Association Model (SAM*), developed for scientific-statistical databases and used for computer-integrated manufacturing applications, provides a set of abstract data types at the lowest level which correspond to the primitive units of the application and provides well defined operations on these units (i.e., sets, vectors, text, and generalized relations)[Su 83]. Such a model has *enhanced* unstructured object representation.

In CAIS attribute kinds are defined in terms of eight basic and structured value types. The type IDENTIFIER is a string having the syntax of an Ada identifier. This type and the basic structured value types such as list, uniform_list, composite_list are accessed only through operations defined on those units.

The IRDS model provides *limited* unstructured object representation. The CAIS-A node model provides *enhanced* unstructured object representation.

A.3.3 Relationship Representation.

Relationships are represented in may ways. Conceptually this construct may appear in the model as an attribute, an entity, tables, an independent element (often called "relationship"), a class or a function. Multiple relationship views may be presented to the user.

In the survey analysis, there is a discussion concerning the fact that the attributes of an entity form the same abstraction as aggregation. Thus, E-R models are characterized as have *tables* and *independent* relationships. However, in the IRDS model there are *data levels.* and therefore the IRDS has *class*, as well as *tables* and *independent* relationship representation.

In CAIS-A, relationships are represented through the *independent* relationships and through *attributes* which do *not* have the same abstraction as aggregation. The CAIS-A model has a *contents* relationship which is quite different from any relationship in a semantic data model and is a reminder that the CAIS-A model is tied to its application, the modeling of an operating system.

A.3.4 Network vs Hierarchy.

This refers to the diagrammatic construct for the conceptualization of a model. The nature of the graph (network vs hierarchy, cyclic versus acyclic) is important in the characterization of the data model. E-R models, including the IRDS, are *strong network*. Semantic networks have a general hierarchy at the generic level and network at the specific level. The CAIS-A model has a general hierarchy at the generic level. Also, the trees of the CAIS-A model present a basically *hierarchical* diagram in the *instance* network.

A.3.5 Derivation/inheritance.

There are two means by which semantic models handle repeated information. Repetition within individual object types is handled by derivation, which is the means by which the attributes of one object are computed or inherited from other objects. Alternatively, class attributes can be used to hold derived information about a class of objects taken as a whole. The IRDS has neither derivation nor inheritance. In CAIS-A there is inheritance at the generic level. At the specific level, CAIS-A has a procedure to *add inheritance*. However adding inheritance to secondary relationships between instances, is like derivation and unlike class inheritance.

A.3.6 Insertion/Deletion/Modification Constraints.

This refers to the constraints that are used to maintain the integrity of the database. Some models permit the database designer to specify the insertion/deletion/modification semantics. Peckham and Maryanski state that insertion/deletion constraints in E-R models are *User specific*. This is probably because there are so many flavors of E-R models. The IRDS has a specific E-R model which has specified the necessary constraints to maintain the integrity of the IRD and IRD Schema Layers. The standard includes a host of error messages, one for each constraint violation. Therefore, this feature of the IRDS model is classified as *Built-in* rather than user-specific. Similarly, the insertion/deletion constraints for nodes in CAIS-A are defined within the standard and *built-in*.

A.3.7 Degree of Expression of Relationship Semantics.

Peckham and Maryanski explain:

> *Some models leave the expression of the semantics of cardinality, null values, inverse relationships, derivations, inheritance, or default values to the designer. Other models completely define the behavior of one or more of these features. The amount of flexibility, and consequently responsibility, given to the designer by the model serves as an important discriminant among models.*

While Peckham and Maryanski state that the relationship semantics of entity-relationship models are *User selectable*, the relationship semantics in the IRDS model are very specific and should be characterized as *Predefined.* In CAIS-A, while the user may specify whether particular relationships are to be bi-directional or inheritable, the relationship semantics are unambiguous, not left to the user, and therefore *Predefined.*

While the relationship semantics are predefined in both IRDS and CAIS-A, the are quite different. Beyond the differences in terminology there seems to be a difference in the *ideas* expressed.

In particular, the word *relationship* in CAIS suggests a concept, such as "foo" which is called a *relationship-class* in the IRDS. The IRDS might have four things called relationships: A-foo-Y, A-foo-X, B-foo-X, B-foo-Y which are grouped together in the relationship-class, foo. These four relationships are *not* instances of foo, they simply a grouping. Relationship-classes and relationships exist at the same data level. Relationship-classes, such as foo, do *not* have instances at the lower level. In the IRDS there are instances only of the relationships, A-foo-X, etc. at the lower level.

In contrast, in CAIS-A "foo" is called a *relationship.* CAIS-A does *not* directly define A-foo-X, etc. at the generic level. The definitions of A, foo, X, etc. do specify the fact that an *instance* of foo will emanate from an instance of A or B and terminate at an instance of X or Y. In CAIS-A there *are* instances of foo. At the moment an instance of a CAIS-A relationship is created its source and target destinations (instances of nodes) are specified. Nodes and paths which appear and disappear as processes are invoked and die provide a model of an operating system, not a semantic data model of persistent entities, their attributes, and their relationships.

A.3.8 Dynamic Modeling.

This term refers to the description of the semantic properties of database transactions. Transaction-modeling is one possibility for dynamic modeling and object-oriented models are another. Dynamic modeling does not apply to the IRDS data model. CAIS-A explicitly supports transactions. We have not determined whether this satisfies the [Peckham 88] definition of dynamic modeling.

A.4 STARS Environment Object Manager

The design of the STARS EOM has begun. In the process of the design work, the Q24 team has investigated SQL, CAIS-A, and the IRDS. The EOM model which was presented in the most recent Q24 deliverable [Q24610B 89] shows that the designers have put the relational model of SQL and the E-R model of the IRDS behind them and that they are stretching toward a model which includes objects, and generalization/specialization hierarchies. The analysis showed that the model being developed is more powerful than the IRDS model. It appears to be a good model for use with its application, the modeling of software engineering environments. It uses *both* of the approaches that have been proposed for adding n-ary relationships to the IRDs model. It is is not clear at this point whether the EOM model will be truly *object-oriented* within the narrow definition which requires both inheritance and specialization of methods.

Our Figure 4 is the first figure in the February 13 Q24 Deliverable [Q24620 89] and shows the mapping of the Objectbase Schema to its underlying implementation on the CAIS-A Node Model. The mapping from the left to the right side of the diagram was extremely useful in considering how an IRDS might be implemented on CAIS-A nodes.

The present analysis is concerned only with the left hand side of the diagram. The next two subsections provide a description of the EOM model and analysis of the model using the Peckham and Maryanski methodology.

A.4.1 Description of the EOM Model

As shown in the figure of the EOM data model, objects are instances of classes. Class definitions define the attributes, methods, and binary relationships to other classes. Methods have arguments, names, algorithm implementation, and a return type/value. Method arguments also have a number of attributes. Also, there is can be a specialization-of/generalization-of between classes. The specialization semantics are not explained in the early design documents, but there is some sort of specialization.

N-ary Relationships are objects. They can have attributes, methods, and (member/member_of) binary relationships to objects. The N-ary relationship is very interesting. It is an *aggregation*, not of its attributes, but of its member objects. The semantics of the relationship-class are not spelled out in the document. Presumably it the *classes* of the member objects (not the objects themselves) which are specified in a relationship-class definition. Instances of the n-ary relationship-class would be an aggregation of objects which are instances of the member classes. It is not stated whether the model allows the classes of member objects to be relationship-classes.

An IRDS proposal on n-ary relationships [X3H489155] discusses two approaches. The approach adopted by the EOM is called a "native" n-ary relationship-type where the n-ary relationship is defined as a schema object. The drawback is that there are now two kinds of binary-relationship, some are like the binary relationships of the IRDS (for example, they don't have methods), and then there would be the 2-ary relationships which can have methods. As Dr. Lefkovitz points out,

- The user has to deal with two different kinds of relationship types.
- Should it ever be necessary to modify a binary relationship-type into an n-ary (even an 2-ary) one, a major modification of the schema would be required.

The alternate approach to the development of n-ary relationships is to allow a (binary) relationship definition which connects an entity and a (binary) relationship, and to allow a binary relationship definition which connects two (binary) relationships. The EOM approach, as depicted, embraces the "native" n-ary approach. However, if it is possible for the members of a relationship-class to be relationship-classes, and why not?, then the second approach is also a part of this model.

A.4.2 Analysis of the EOM Model

The EOM Model which has been proposed by the Boeing Q24 Task is analyzed using the eight characteristics specified in [Peckham 88].

- **Unstructured object representation.** As explained earlier, the CAIS-A node model provides for *enhanced* unstructured object representation in the definition of attribute kinds. This characteristic will be available to the EOM which uses CAIS-A nodes.

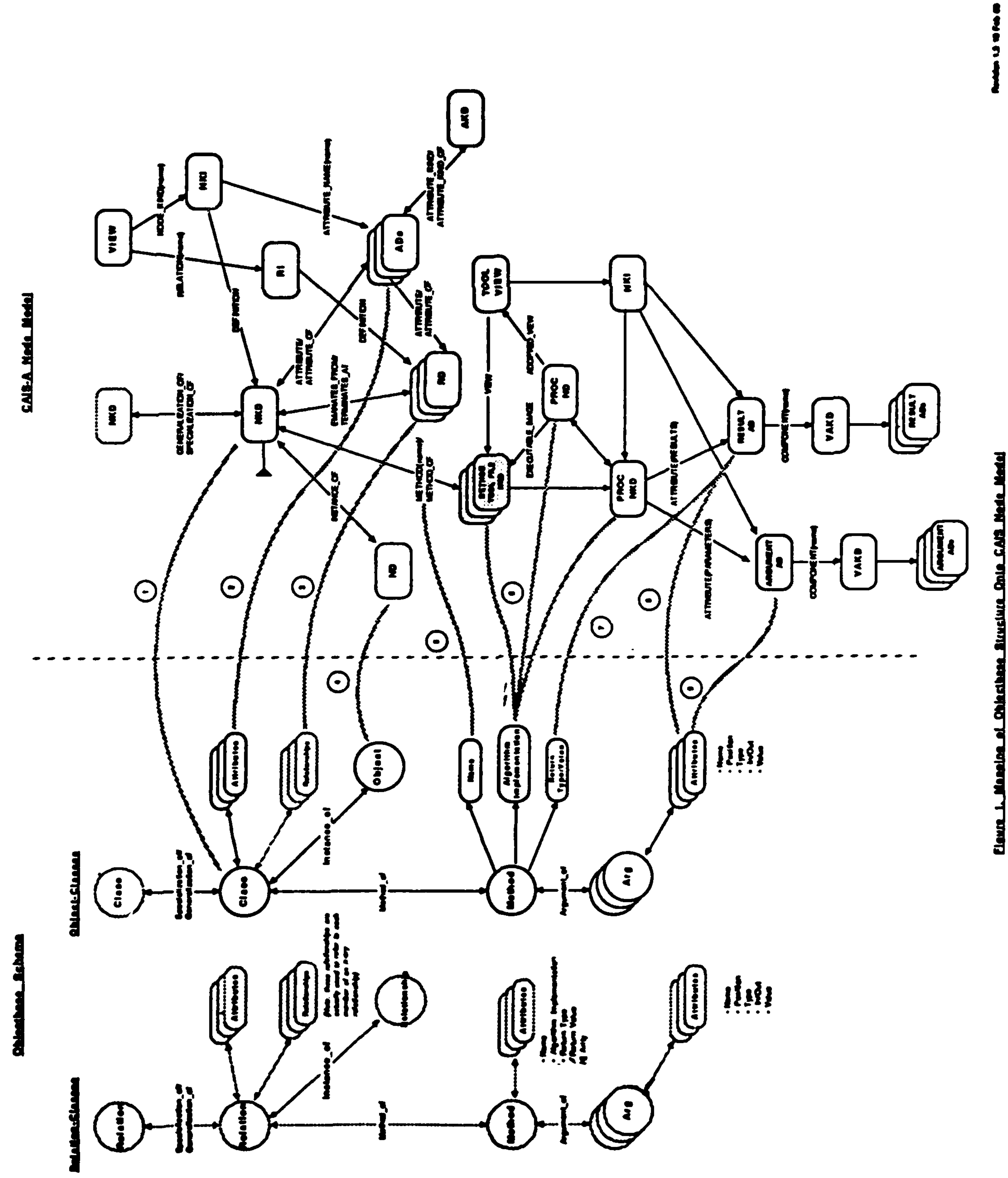

Figure 4. Boeing Q24 EOM Model
Object Base Structure Definition
D613-100620

- **Relationship Representation.** Relationships are represented in an wide variety of ways. Through *attributes, binary relationships, n-ary relationships, and properties* The n-ary relationship is an entity with three different kinds of properties: attributes, members, and methods. Moreover, through the implementation on CAIS-A NKDs, some objects will also have contents.

- **Standard Abstractions Present.** All four abstractions are present: Generalization, Aggregation, Classification, Association The first three are inherent in the model, the fourth is provided by the Object Manipulation Language which is described in a companion Q24 Task deliverable, [Q24620 89].

- **Network vs Hierarchy.** *Strong network, General hierarchy present.* Specialization semantics are not yet well defined, so it is not yet clear whether the model will meet the narrow definition of object-oriented which was given above. With strong specialization semantics then the EOM would be characterized as having a *strong hierarchy.*

- **Derivation/inheritance.** *Derivation and Inheritance.*

- **Insertion/Deletion/Modification Constraints.** To what extent are constraints *built in* and can *user specific* constraints be introduced? Constraint issues are not discussed in the design document, however certain *built-in* constraints of the CAIS-A node model would be reflected in the EOM model. Other built-in constraints might also become part of the model.

  In general, a mapping can be developed from the EOM model to the model of semantic networks. Some semantic networks, i.e., PCLnet [Heineman 87], have *user-defined* constraints which appear in the diagramatic representations of the model. Others, i.e., AdaKnet [Wallnau 88], do not. The diagram of the EOM model does *not* show constraints. An earlier design document [Q24570 88] does state that the OBMS must support the concept of *active data* which is functionally defined data, accessed through an object's interface and providing for the maintenance of constraints among objects. This statement suggests that *user-defined* constraints are contemplated.

- **Degree of Expression of Relationship Semantics.** These should be predefined in the model. Judging by early design documents, issues of relationship semantics, in particular n-ary relationship semantics and specialization semantics, need to be addressed and clarified by the designers.

- **Dynamic Modeling.** The document on the Object Manipulation Language indicates that the EOM will have *transaction modeling.* As indicated in the section on constraints, the EOM model will support *active data.* It is not yet clear whether it will be *object-oriented* within the limits of the narrow definition provided above, i.e., provide a capability for specialization of inherited methods.

The EOM model goes far beyond the EA/RA model of the IRDS. It provides generalization/specialization, objects, n-ary relationships, and a capability for unstructured object representation. The following excerpt from [Winkler 88] is apropos:

> *The discussion . . .reminds me of heated discussions in the early 1970s between E. F. Codd and Charles Bachman, the "fathers" of the relational and network data*

models, respectively. In these discussions, one father defended the old religion, while the other proposed the benefits and validity of the new religion. I believe that eighteen years later we may be at a new crossroads. At this crossroad, one discovers the old religion, SQL, and a potential new religion, E-R. Of course, one also needs to understand the relationship between E-R and object-oriented, for it may be that a melding of these two technologies could form an even more long-lived religion.

References

[AMS 88] AMS. The Ada Environment for Requirements Analysis, Conceptual Design, and Prototyping of the Ada Subset of the Army Data Encyclopedia. August 1988. American Management Systems; Deliverable 3.1 of Contract No. GS-00K-85-AFD-2777.

[ANSI 88] ANSI. *American National Stndard X3-138-1988.* American National Standards Institute, New York, 1988.

[AOG 83a] AOG. *The Ada SYSTEM-STANDARD SCHEMA.* 1983. Prepared for the AJPO under NBS Contract NB81SBCA0735.

[AOG 83b] AOG. *Final Report on the Applicability of the FIPS DDS to Ada Programming Support Environments.* 1983. Prepared for the AJPO under NBS Contract NB81SBCA0735.

[Batini 86] Batini, C. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys* 18(4), December 1986.

[CAISA 88] CAIS-A. *Common Ada Programming Support Environment (APSE) Interface Set (CAIS) (Revision A).* 1988. Proposed DOD-STD-1838A.

[Dwyer 88] Dwyer, P., Hou, C., Kumar, A., and Larson, J. *ANSWER: Army's Nonprogrammer System.* Honeywell, Inc., 1988. Funded under AIRMICS Contract No. DAKF11-88-C-0024.

[Goldfine 88a] Goldfine, A. *Proceedings of the FIPS Workshop on IRDS Applications.* National Institute of Standards and Technology, Gaithersburg, MD, 1988. NISTIR 88-3896.

[Goldfine 88b] Goldfine, A. *Using the Information Resource Dictionary System Command Language (Second Edition).* National Bureau of Standards, Gaithersburg, MD, 1988. NBSIR 88-3701.

[Goldfine 88c] Goldfine, A. and Konig, P. *A Technical Overview of the Information Resource Dictionary System (Second Edition).* National Bureau of Standards, Gaithersburg, MD. January,, 1988. NBSIR 88-3700.

[Heineman 87] Heineman, M., McKay, D., Fritzson, R., and Seavey, B. *PCLnet - Version 1.0.* Technical Report 65, Logic Based Systems, Paoli Research Center, Unisys, 1987.

[Hull 87] Hull, R. and King, R. Semantic Database Modeling: Survey, Applications, and Research Issues. *ACM Computing Surveys* 19(3), September 1987.

[Law 88] Law, M. H. *Guide to IRDS Applications: General Concepts and Strategic Systems Planning.* National Bureau of Standards, Gaithersburg, MD, 1988. NBS Special Publication 500-152.

[LBL 88] LBL. *Data Encyclopedia Architecture for Army Information Management.* Lawrence Berkeley Laboratories, June,, 1988. Prepared for the USAISEC under DOE Contract De-AC03-76SF0098.

[McCarthy 88] McCarthy, J. The Automated Data Thesaurus: A New Tool for Scientific Information. In *Proceedings of the 11th International Codata Conference.* Karlsruhe, Germany, September 1988.

[McKay 87] McKay, C., Charette, R., and Auty, D. *A Study to Identify Tools Needed to Extended the MAPSE to Support the Life Cycle of Large, Complex, Non-Stop, Distributed Systems Such as the Space Station Program.* 1987. Contract No. NAS 9-17010.

[McKay 88a] McKay, C. *A 2 A.M. Position Paper on the Use of the Current Version of POSIX for the Target Environment of the Space Station Program.* 1988.

[McKay 88b] McKay, C. A Framework for Managing the Life Cycle Reuse of Software Engineering Products and Process. 1988. 48 slides for a talk.

[McKay 88c] McKay, C. *A Position Paper: The Need to Advance Beyond the Relational Data Model and SQL.* Technical Report, University of Houston Clear Lake, January 1988.

[McKay 89] McKay, C. Conceptual and Implementation Models Which Support Life Cycle Reusability of Processes and Products in Computer Systems and Software Engineering. In *Ada Reusability Guidelines*, Chapter III. 1989. AIRMICS - HTL Grant 2-5-51537.

[Newton 87] Newton, J. *Guide on Data Entity Naming Conventions.* National Bureau of Standards, Gaithersburg, MD, 1987. NBS Special Publication 500-149.

[Peckham 88] Peckham, J. and Maryanski, F. Semantic Data Models. *ACM Computing Surveys* 20(3), September 1988.

[Q24570 88] Q24/570. Functional Capabilities Review for OBMS. December 1988. Boeing STARS Environment Object Manager task.

[Q24590A 89] Q24/590A. ODL, OML, and Object Dictionary of the OBMS. January 1989. Boeing STARS Environment Object Manager task.

[Q24610B 89] Q24/610B. EOM Prototype I, Object Manipulation Language Design. February 1989. Boeing STARS Environment Object Manager task.

[Q24620 89] Q24/620. Object Base Structure Definitions. February 1989. Boeing STARS Environment Object Manager task.

[Robinson 88] Robinson, G. Argonne National Laboratory. In *Proceedings of the FIPS Workshop on IRDS Applications.* Gaithersburg, MD, December 1988. NBSTIR 88-3896.

[SAA 88] SAA. Systems Application Architecture. *IBM Systems Journal* 27(3), 1988. entire issue.

[SC21N2642 88] SC21/N2642. *IRDS Framework.* ISO/IEC JTC1, March, 1988.

[SC21WG3N669 88] SC21/WG3/N669. *IRDS Services Interface, Working Draft, Revision 7.* ISO/IEC JTC1, September, 1988.

[SofTech 88] SofTech. *CAIS READER'S GUIDE for DOD-STD-1838A (DRAFT).* 1988. Prepared by SofTech for AJPO, NOSC Contract No. N66001-86-D-0101.

[Su 83] Su, S. W. Y. SAM*: A semantic association model for corporate and scientific-statistical databases. *Inf. Sci.* 29, 1983.

[Wallnau 88] Wallnau, K., Solderitsch, J., Simos, M., McDowell, R., Cassell, K., and Campbell, D. Construction of Knowledge-Based Components and Applications in Ada. In *Proceedings of AIDA-88, Fourth Annual Conference on Artificial Intelligence & Ada.* Fairfax, Virginia, November 1988.

[Winkler 88] Winkler, J. The Entity-Relationship Approach and the Information Resource Dictionary System Standard. In *Seventh International Conference on Entity-Relationship Approach.* Rome, Italy, November 1988. Keynote Address by the Chair of ANSC X3H4; X3H4/89-027.

[X3H4 88] X3H4. *IRDS Services Interface (draft proposed).* X3 Secretariat, CBEMA, Washington, D.C., 1988.

[X3H489016] X3H4/89-016. Proposal to produce a technical report, IRDS support of model integration. January, 1989. Source: Harry Goldberg, Bellcore.

[X3H489155] X3H4/89-155. N-ary Relationship-Types. November, 1988. Source: Henry Lefkovitz, AOG.